YILDIZ TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SENIOR PROJECT

FLAME RECOGNITION IN VIDEO

Project Supervisor: Prof. Dr. A. Coşkun Sönmez

Project Group

02011040 Eren Aykın

Istanbul, 2007

CONTENTS

SYMBOL LIST

Fps	Frames per Second
Hz	Herz

ABBREVIATION LIST

FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
CAMShift	Complex Adaptive MeanShift
ROI	Region of Interest

FIGURE LIST

TABLE LIST

FOREWORD

I would like to thank to my project supervisor Prof. Dr. A. Coskun Sonmez for his help and support during the project.

ABSTRACT

Conventional fire detection systems are generally limited to indoors and they usually fail when detection in open area needed. A system that is able to detect fire by processing real-time video images would both work in open area and it doesn't need a high budget. It can also be used in accordance with surveillant cameras for better performance. This type of detection systems can provide improved detection and fewer false alarms since they detect the combustion itself instead of its byproducts. Additional descriptive information about fire location and size is possible, and these can be useful for preventing fires' scattering.

In this project, we propose 3 methods for flame recognition in video images using flame's motion, color and flickering properties, and provide a comparison between the efficiency and success of those methods.

1. INTRODUCTION

Most of the proposed methods and conventional systems used in today's areas that need fire protection, are for indoors, and their mechanical systems are designed to detect not the fire itself but its byproducts. Presence of certain particles generated by smoke and fire is detected by most of those systems. Alarm is not issued unless particles reach the sensors to activate them. Also, infrared and ultraviolet sensors that are also commonly used produce many false alarms. By the help of machine vision techniques, it is possible to get better results than conventional systems because images can provide more reliable information.

1.1. Related Work

Early detection of fire is one of the humanities oldest and most important problems, therefore there have been many methods proposed to solve this issue. However, detecting fire from video images is a relatively new subject. Some of the related work in areas of machine learning and image processing about this subject are as follows:

B. Uğur Töreyin, Yiğithan Dedeoğlu, Uğur Güdükbay, A. Enis Çetin
Computer vision based method for real-time fire and flame detection [1]

Flame and fire flicker is detected by analyzing the video in the wavelet domain.

Spatial wavelet analysis that is used in this work makes it possible to detect high frequency behavior not only on the contours of the image that is generally the only part taken into account, but also inside the fire region area. A fire colored moving object doesn't cause changes in the values of wavelet coefficients because there is no variation in fire colored pixel values, but the activity within the fire region should produce variation in energy of wavelet coefficients.

Using this rule in the project, Quasi-periodic behavior in flame boundaries is detected by performing temporal wavelet transform. Color variations in flame regions are detected by computing the spatial wavelet transform of moving fire-colored regions.

G. Healey, D. Slater, T. Lin, B. Drda, and D. Goedeke.
A system for real-time fire detection [2]

Algorithms for fire detection based on the spectral, spatial, and temporal properties of fire events are developed.

Video image is firstly cut into grids, and possible fire pixels are labeled as connected components. Following connected component analysis, the event interpretation component of the system measures event growth using previous event data.

This project uses a purely color based model which uses characteristics that has been derived from large amount of video data.

Nobuyuki Fujiwara, Kenji Terada,
Extraction of a Smoke Region Using Fractal Cording [3]

This project proposes a technique for extracting smoke regions from an image using fractal encoding concepts. Smoke shapes have the properly of self-similarity, and the project attempts to extract smoke regions by discovering the distinguishing features of smoke regions in the code produced by fractal encoding of an image.

Wen-Bing Homg, Jim-Wen Peng, and Chih-Yuan Chen
A New Image-Based Real-Time Flame Detection Method Using Color Analysis [4]

Fire flame features based on the HSI color model are extracted by analyzing 70 flame images. Then, based on these flame features, regions with fire-like colors are roughly separated from an image by color separation method. Then, the image difference method and the invented color masking technique based on chromatics are used to

remove spurious fire-like regions, such as objects with similar fire colors or areas reflected from fire flames. Finally, the burning degree of fire flames is estimated using the fact that hotter fires (200-280 Degrees C) have more blue color while lower temperature causes the color turn into yellow from red (0-60 Degrees C).

Two reasons for using the HSI color model in the project are: First, the intensity component is decoupled from the color information in an image. Second, the hue and saturation components are intimately related to the way in which human beings perceive color.

Steps:

$$i = \frac{1}{3}(r+g+b) \quad (1)$$

$$s = 1 - \frac{3}{(r+g+b)}[\min(r,g,b)] \quad (2)$$

$$h = \begin{cases} \theta & \text{if } b \le g \\ 360-\theta & \text{if } b > g \end{cases} \quad (3)$$

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{1/2}}\right\} \quad (4)$$

Walter Phillips III, Mubarak Shah, Niels da Vitoria Lobo
Flame Recognition in Video [5]

The proposed method in the article can cope with problems caused by moving cameras or moving scenes. It first uses an approach that is based upon creating a Gaussian-smoothed color histogram to detect the fire-colored pixels, and then uses a temporal variation of pixels to determine which of these pixels are fire pixels. Next, some spurious fire pixels are automatically removed using an erode operation, and some missing fire pixels are found using region growing method. The method used to make it insensitive for camera or scene motion is, to subtract non-fire pixel temporal variance from fire-pixel temporal variance.

Robert T. Collins, Alan J. Lipton and Takeo Kanade
A System for Video Surveillance and Monitoring [6]

Presents an overview of the system being developed and methods used by the Robotics Institute at Carnegie Mellon University (CMU) and the Sarnoff Corporation in order to provide a continuous coverage of people and vehicles in cluttered environments.

Che-Bin Liu, Narendra Ahuja
Vision Based Fire Detection [7]

Spectral, spatial and temporal models of fire regions in visual image sequences are presented. The spectral model is represented in terms of the color probability density of fire pixels. The spatial model captures the spatial structure within a fire region. The shape of a fire region is represented in terms of the spatial frequency content of the region contour using its Fourier coefficients. The temporal changes in these coefficients are used as the temporal signatures of the fire region. Specifically, an autoregressive model of the Fourier coefficient series is used.

B. Uğur Töreyin, Yiğithan Dedeoğlu, A. Enis Çetin
Wavelet Based Real-Time Smoke Detection in Video [8]

The monitoring camera is assumed to be stationary. When smoke is present, edges of image frames start loosing their sharpness and this leads to a decrease in the high frequency content of the image. The background of the scene is estimated and decrease of high frequency energy of the scene is monitored using the spatial wavelet transforms of the current and the background images. Edges of the scene produce local extrema in the wavelet domain. A decrease in values of local extrema is also an indicator of smoke. Also scene becomes grayish
when there is smoke and this leads to a decrease in chrominance values of pixels.

J. Huseynov, Z. Boger, G. Shubinsky, S. Baliga
Optical Flame Detection Using Large-Scale Artificial Neural Networks [9]

A model for hydrocarbon flame detection using artificial neural networks with over 1000 input features is used in the project. Joint time-frequency analysis in the form of Short-Time Fourier Transform is used for extracting the relevant features from infrared sensor signals. After appropriate scaling, this information is provided as an input for the ANN training algorithm based on conjugate-gradient descent method. A classification scheme with trained ANN connection weights is implemented on a digital signal processor for an industrial hydrocarbon flame detector.

1.2. Fire and Its Properties

Typically, fire comes from a chemical reaction between oxygen in the atmosphere and some sort of fuel. For the combustion reaction to happen, fuel must be heated to its ignition temperature. In a typical wood fire: First something heats the wood to a very high temperature. When the wood reaches about 150 degrees Celsius, the heat decomposes some of the cellulose material that makes up the wood. Some of the decomposed material is released as volatile gases. We know these gases as smoke. Smoke is compounds of hydrogen, carbon and oxygen. The rest of the material forms char, which is nearly pure carbon, and ash, which is all of the unburnable minerals in the wood (calcium, potassium, and so on). The char is also called charcoal. Charcoal is wood that has been heated to remove nearly all of the volatile gases and leave behind the carbon. That is why a charcoal fire burns with no smoke.

The actual burning of wood then happens in two separate reactions: (1) When the volatile gases are hot enough (about 260 degrees C for wood), the compound molecules break apart, and the atoms recombine with the oxygen to form water, carbon dioxide and other products. In other words, they burn. (2) The carbon in the char combines with oxygen as well, and this is a much slower reaction. A side effect of these chemical reactions is a lot of heat. The fact that the chemical reactions in a fire generate a lot of new heat is what sustains the fire.

As they heat up, the rising carbon atoms (as well as atoms of other material) emit light. This "heat produces light" effect is called incandescence. It is what causes the visible flame. Flame color varies depending on what is being burned and how hot it is. Color variation within in a flame is caused by uneven temperature. Typically, the hottest part of a flame glows blue, and the cooler parts at the top glow orange or yellow.
The dangerous thing about the chemical reactions in fire is the fact that they are self-perpetuating. The heat of the flame itself keeps the fuel at the ignition temperature, so it continues to burn as long as there is fuel and oxygen around it. The flame heats any surrounding fuel so it releases gases as well. When the flame ignites the gases, the fire spreads.

On Earth, gravity determines how the flame burns. All the hot gases in the flame are much hotter (and less dense) than the surrounding air, so they move upward toward lower pressure. This is why fire typically spreads upward, and it's also why flames are always "pointed" at the top.

2. SYSTEM ANALYSIS AND FEASABILITY STUDY

Video processing causes a lot of overhead on the computer, so a powerful CPU and a large enough memory must be used.

2.1. Structure of the Current System

Processor: 1100 MHz
Memory: 512 MB SD RAM
Hard disk: 80 GB

2.2. Software Tools Used

Microsoft Visual C++ Express Edition
OpenCV Library
Trolltech QT4 GUI Library

2.3. Gantt Chart

	Week											
Tasks	1	2	3	4	5	6	7	8	9	10	11	12
Planning												
Definition of the Problem												
Preparation of the Hardware & Software												
Coding												
Test												
Mile Stones												
1. Report						**X**						
Software Presentation										**X**		
2. Report												**X**

Table 2.1. Gantt Chart

2.4. Cost of the Project

2.4.1 Software

Microsoft Visual C++ Express Edition: 0$

Trolltech QT4 GUI Library (Open Source Edition): 0$

Total: 0$

2.4.2 Hardware

PC: 500$

Web Cam: 50$

Total: 550$

2.4.3 Software Development

Development time: 12 Weeks
Development cost per week: 50$
Total = 600$

2.4.4 Total Cost

0$ + 550$ + 600$ = 1150$

2.5 Open Source Computer Vision Library (OpenCV)

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision. Example applications of the OpenCV library are Human-Computer Interaction (HCI); Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, Motion Understanding; Structure From Motion (SFM); and Mobile Robotics. The OpenCV Library software package supports many functions whose performance can be significantly enhanced on the Intel® architecture (IA), particularly...

The OpenCV Library is a collection of low-overhead, high-performance operations performed on images. The OpenCV implements a wide variety of tools for image interpretation. It is compatible with Intel® Image Processing Library (IPL) that implements low-level operations on digital images. In spite of primitives such as binarization, filtering, image statistics, pyramids, OpenCV is mostly a high-level library implementing algorithms for calibration techniques (Camera Calibration), feature detection (Feature) and tracking (Optical Flow), shape analysis (Geometry, Contour Processing), motion analysis (Motion Templates, Estimators), 3D reconstruction (View Morphing), object segmentation and recognition (Histogram, Embedded Hidden

Markov Models, Eigen Objects). The essential feature of the library along with functionality and quality is performance. The algorithms are based on highly flexible data structures (Dynamic Data Structures) coupled with IPL data structures; more than a half of the functions have been assembler-optimized taking advantage of Intel® Architecture (Pentium® MMX, Pentium® Pro, Pentium® III, Pentium® 4).

The OpenCV Library is a way of establishing an open source vision community that will make better use of up-to-date opportunities to apply computer vision in the growing PC environment. The software provides a set of image processing functions, as well as image and pattern analysis functions. The functions are optimized for Intel® architecture processors, and are particularly effective at taking advantage of MMX® technology. The OpenCV Library has platform-independent interface and supplied with whole C sources.

2.6 Trolltech QT4

Qt by Trolltech is a C++ toolkit for cross-platform GUI application development. Qt provides single-source portability across Microsoft Windows, Mac OS X, Linux, all major commercial Unix variants, and embedded Linux. On embedded Linux, the Qt API is available as Qt/Embedded. Qt provides application developers with all the functionality needed to build applications with state-of-the-art graphical user interfaces. Qt is fully object-oriented, easily extensible, and allows true component programming. Since its commercial introduction in early 1996, Qt has formed the basis of many thousands of successful applications worldwide. Qt is also the basis of the popular KDE Linux desktop environment, a standard component of all major Linux distributions. See our Customer Success Stories for some examples of commercial Qt development.

Qt is supported on the following platforms:

- Microsoft Windows -- 98, NT 4.0, ME, 2000, and XP
- Unix/X11 -- Linux, Sun Solaris, HP-UX, HP Tru64 UNIX, IBM AIX, SGI IRIX and many others
- Mac OS X -- Mac OS X 10.2+

- Embedded Linux -- Linux platforms with framebuffer support

Qt is released in different editions:

- The Qt Commercial Editions are used for commercial software development. They permit traditional commercial software distribution and include free upgrades and technical support. For the latest prices, see the online Pricing Information page or contact sales@trolltech.com.
- The Qt Open Source Edition is for the development of Free and Open Source software only. It is provided free of charge under the terms of both the Q Public License and the GNU General Public License.

With the release of Qt 4, Trolltech Inc. has added Visual Studio .NET integration to its C++ application development framework. Qt lets developers write an application once and use the same code base across operating systems, so VS.NET integration means Windows developers can create apps to run on Unix, Linux and Mac OS platforms. Other new features in Qt 4 include advanced multi-threading capabilities, database integration, XML support and an improved graphics subsystem.

2.7. FFTW Discrete Fourier Transform (DFT) Library

FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST).

Benchmarks, performed on on a variety of platforms, show that FFTW's performance is typically superior to that of other publicly available FFT software, and is even competitive with vendor-tuned codes. In contrast to vendor-tuned codes, however, FFTW's performance is *portable*: the same program will perform well on most architectures without modification. Hence the name, "FFTW," which stands for the somewhat whimsical title of "Fastest Fourier Transform in the West."

The FFTW package was developed at MIT by Matteo Frigo and Steven G. Johnson.

FFTW 3.1.2 is the latest official version of FFTW. Here is a list of some of FFTW's more interesting features:

- Speed. (Supports SSE/SSE2/3dNow!/Altivec, since version 3.0.)
- Both one-dimensional and multi-dimensional transforms.
- Arbitrary-size transforms. (Sizes with small prime factors are best, but FFTW uses O(N log N) algorithms even for prime sizes.)
- Fast transforms of purely real input or output data.
- Transforms of real even/odd data: the discrete cosine transform (DCT) and the discrete sine transform (DST), types I-IV. (Version 3.0 or later.)
- Efficient handling of multiple, strided transforms. (This lets you do things like transform multiple arrays at once, transform one dimension of a multi-dimensional array, or transform one field of a multi-component array.)
- Parallel transforms: parallelized code for platforms with Cilk or for SMP machines with some flavor of threads (e.g. POSIX). An MPI version for distributed-memory transforms is also available, currently only as part of FFTW 2.1.5.
- Portable to any platform with a C compiler. Documentation in HTML and other formats.
- Both C and Fortran interfaces.

3. POJECT DETAILS

There will be used 3 methods for flame detection in the project. While the first method, first detects motion in the video and then checks if moving pixels have the color of flame, the second method first detects the contours of the images by edge detection. If colors of the most of the pixels belonging to these contours have flame color properties, it is checked if they are in the motion detected area of the image by using the CAMShift Algorithm. If so, the area circled by the contour is marked as flame. In the last method by the help of Fourier Transform, it is looked for if the frequency of average values of fire colored contour pixels are changing with a frequency close to 10Hz, which is an indicator of a flame.

3.1. Frame Subtraction Method for Flame Detection

3.1.a Motion Detection

System looks for if there is motion going on in consequent frames. The method proposed for motion detection is Frame Subtraction. By subtracting consequent frames from each other, the area where motion occurs can be detected. In order to remove false detections caused by motion of the camera, the difference of pixels belonging to areas outside of the motion can also be calculated and then subtracted from all pixels. This method removes the necessity of using a stable camera.

Figure 3-1. Detection of motion with frame subtraction

3.1.b Color Detection

Consequent pixels that belong to a flame have certain properties. According to a previous work [1], A sample fire-color cloud in RGB space (a), and the spheres centered at the means of the Gaussian distributions with radius twice the standard deviation (b) are shown in Figure 3-2.

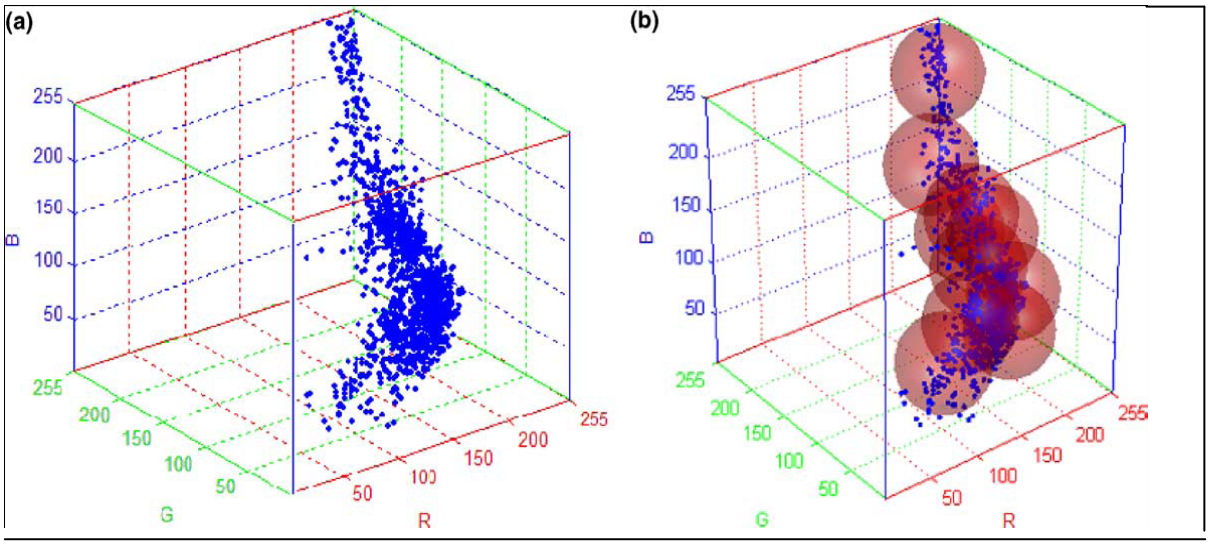

Figure 3-2. Fire-pixel properties in the RGB Space [1]

Fire and flame color model proposed in [9] defines flame pixels. In the RGB space, there must be a relation between the R, G and B color channels that is: R > G > B. R has to have a higher value than a pre-determined threshold RT. Since lighting conditions in the background can cause non-flame pixels look like flame pixels, Saturation values of the pixels under inspection must also has a higher value than some pre-determined threshold ST. ST is the value of saturation when the value of R channel is *RT*. So the conditions we have for a flame pixel are:

1. R > RT
2. R > G > B
3. S > (255 – R) * ST / RT

3.1.c Advantages of the Method

- It is easy to implement the method
- Every pixel in the frame is checked
- Doesn't need a stable camera

3.1.d Disadvantages of the Method

- Needs more processing power for it's calculations for each frame of each pixel

3.2 CAMShift Method for Flame Detection

This method uses the same color recognition method, but it improves the motion detection by tracking the moving area of the video frames and making calculations only in that area.

3.2.a Segmentation and Contour Detection

Since areas that belong to fire in a video frame that is turned into gray level will have color values close to white, by a segmentation method like Canny algorithm can be used to segment those areas. Once the whitish areas are detected, the information of coordinates of the pixels that belong to the borders of those segments can be attained. In order to reduce the possibility of selecting wrong segments, it can be checked if half of the contour pixels has the color of flame by using the formula which was used in the first method.

3.2.a.1 Canny Edge Detection Algorithm

The Canny operator was designed to be an optimal edge detector (according to particular criteria there are other detectors around that also claim to be optimal with respect to slightly different criteria). It takes as input a grey scale image, and produces as output an image showing the positions of tracked intensity discontinuities.

The Canny operator works in a multi-stage process. First of all the image is smoothed by Gaussian convolution. Then a simple 2-D first derivative operator is applied to the smoothed image to highlight regions of the image with high first spatial derivatives. Edges give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as *non-maximal suppression*. The tracking process exhibits hysteresis controlled by two thresholds: *T1* and *T2* with *T1* > *T2*. Tracking can only begin at a point on a ridge higher than *T1*. Tracking then continues in both directions out from that point until the height of the ridge falls below *T2*. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments.

The effect of the Canny operator is determined by three parameters: The width of the Gaussian mask used in the smoothing phase, and the upper and lower thresholds used by the tracker. Increasing the width of the Gaussian mask reduces the detector's sensitivity to noise, at the expense of losing some of the finer detail in the image. The

localization error in the detected edges also increases slightly as the Gaussian width is increased.

Usually, the upper tracking threshold can be set quite high, and the lower threshold quite low for good results. Setting the lower threshold too high will cause noisy edges to break up. Setting the upper threshold too low increases the number of spurious and undesirable edge fragments appearing in the output.

The higher intensity gradients are more likely to be edges. There is not an exact value at which a given intensity gradient switches from not being an edge into being an edge. Therefore Canny uses thresholding with hysteresis. Thresholding with hysteresis requires two thresholds - high and low. Making the assumption that important edges should be in continuous lines through the image allows us to follow a faint section of a given line, but avoid identifying a few noisy pixels that do not comprise a line. Therefore we begin by applying a high threshold. This marks out the edges we can be fairly sure are genuine. Starting from these, using the directional information derived earlier, edges can be traced through the image. While tracing a line, we apply the lower threshold, allowing us to trace faint sections of lines as long as we find a starting point.
Once this process is complete we have a binary image where each pixel is marked as either an edge pixel or a non-edge pixel.
A more refined approach to obtain edges with sub-pixel accuracy is by detecting zero-crossings of the second-order directional derivative in the gradient direction (Lindeberg 1998)

$$L_x^2\, L_{xx} + 2\, L_x\, L_y\, L_{xy} + L_y^2\, L_{yy} = 0,$$

that satisfy a sign-condition on the third-order directional derivative in the gradient direction

$$L_x^3\, L_{xxx} + 3\, L_x^2\, L_y\, L_{xxy} + 3\, L_x\, L_y^2\, L_{xyy} + L_y^3\, L_{yyy} < 0$$

where L_x, L_y ... L_{yyy} denote partial derivatives computed from a scale-space representation L obtained by smoothing the original image with a Gaussian kernel. By definition the edge segments obtained in this way will be continuous curves, so no refined edge tracking is necessary. Hysteresis thresholding can also be applied to these subpixel edges.

One problem with the basic Canny operator is to do with Y-junctions *i.e.* places where three ridges meet in the gradient magnitude image. Such junctions can occur where an edge is partially occluded by another object. The tracker will treat two of the ridges as a single line segment, and the third one as a line that approaches, but doesn't quite connect to, that line segment.

Figure 3-3. Canny Edge detection is used for finding contours

3.2.b Motion Detection

There can be other images with their contours having fire pixel colors. We also need to make sure that these fire colored contours have motion. In order to detect motion, Continuously Adaptive Meanshift algorithm is used. Since this algorithm needs initial selection of the object to be tracked, the initial window of tracking is determined by the help of the method we have used for contour and color detection. A large enough (100-160 pixels long) rectangular area of a fire contour is selected as the initial tracking window and it is checked if the fire colored contours found with the previous method belong to the area where motion is detected.

3.2.b.1 CAMShift Algorithm

CamShift stands for the "Continuously Adaptive Mean-SHIFT" algorithm. Figure 3 summarizes this algorithm. For each video frame, the raw image is converted to a color probability distribution image via a color histogram model of the color being tracked, e.g., flesh color in the case of face tracking. The center and size of the color object are found via the CamShift algorithm operating on the color probability image. The current size and location of the tracked object are reported and used to set the size and location of the search window in the next video image. The process is then repeated for continuous tracking. The algorithm is a generalization of the Mean Shift algorithm, highlighted in gray in Figure 3.

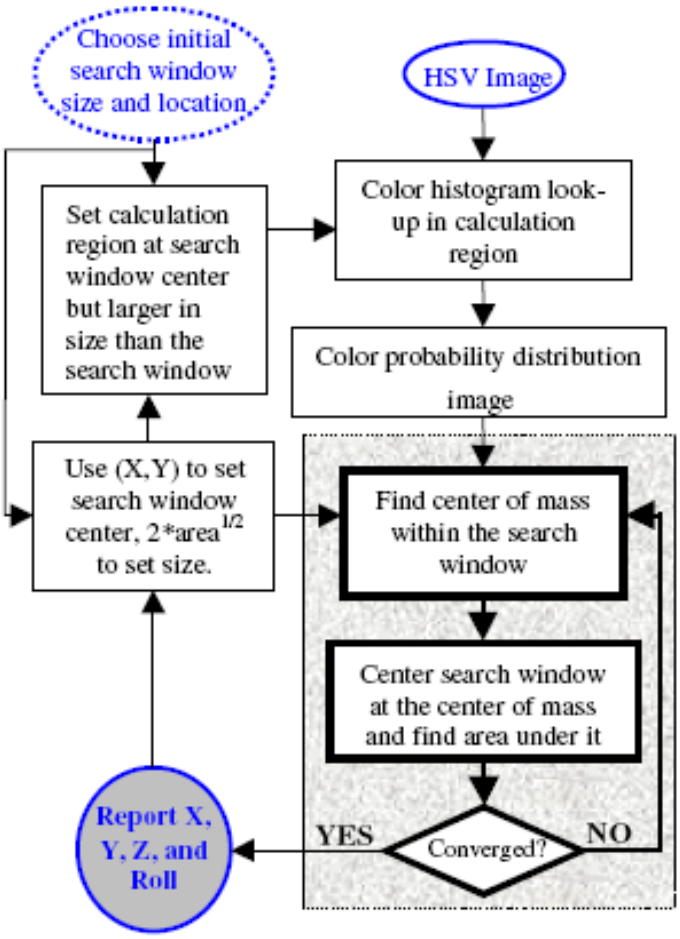

Figure 3-4. The CAMShift Algorithm

CamShift operates on a *2D* color probability distribution image produced from histogram back-projection. The core part of the CamShift algorithm is the Mean Shift algorithm. The Mean Shift part of the algorithm (gray area in Figure 3) is as follows:

1. Choose the search window size.
2. Choose the initial location of the search window.
3. Compute the mean location in the search window.
4. Center the search window at the mean location computed in Step 3.
5. Repeat Steps 3 and 4 until the search window center converges, i.e., until it has moved for a distance less than the preset threshold.

3.2.b.1.a Mass Center Calculation for 2D Probability Distribution

For discrete *2D* image probability distributions, the mean location (the centroid) within the search window, that is computed at step 3 above, is found as follows:
Find the zeroth moment

$$M_{00} = \sum_x \sum_y I(x, y) .$$

Find the first moment for *x* and *y*

$$M_{10} = \sum_x \sum_y xI(x, y) ; M_{01} = \sum_x \sum_y yI(x, y) .$$

Mean search window location (the centroid) then is found as

$$x_c = \frac{M_{10}}{M_{00}} ; y_c = \frac{M_{01}}{M_{00}} ,$$

where $I(x,y)$ is the pixel (probability) value in the position (x,y) in the image, and x and y range over the search window.

Unlike the Mean Shift algorithm, which is designed for static distributions, CamShift is designed for dynamically changing distributions. These occur when objects in video sequences are being tracked and the object moves so that the size and location of the probability distribution changes in time. The CamShift algorithm adjusts the search window size in the course of its operation. Initial window size can be set at any reasonable value. For discrete distributions (digital data), the minimum window length or width is three. Instead of a set, or externally adapted window size, CamShift relies on the zeroth moment information, extracted as part of the internal workings of the algorithm, to continuously adapt its window size within or over each video frame.

Steps in order to detect motion by Camshift algorithm are as follows:

1. Set the calculation region of the probability distribution to the whole image.
2. Choose the initial location of the *2D* mean shift search window.
3. Calculate the color probability distribution in the *2D* region centered at the search window location in an ROI slightly larger than the mean shift window size.
4. Run Mean Shift algorithm to find the search window center. Store the zeroth moment (area or size) and center location.
5. For the next video frame, center the search window at the mean location stored in Step 4 and set the window size to a function of the zeroth moment found there. Go to Step 3.

Figure 3-4 shows CamShift finding the face center on a *1D* slice through a face and hand flesh hue distribution. Figure 3-5 shows the next frame when the face and hand flesh hue distribution has moved, and convergence is reached in two iterations.

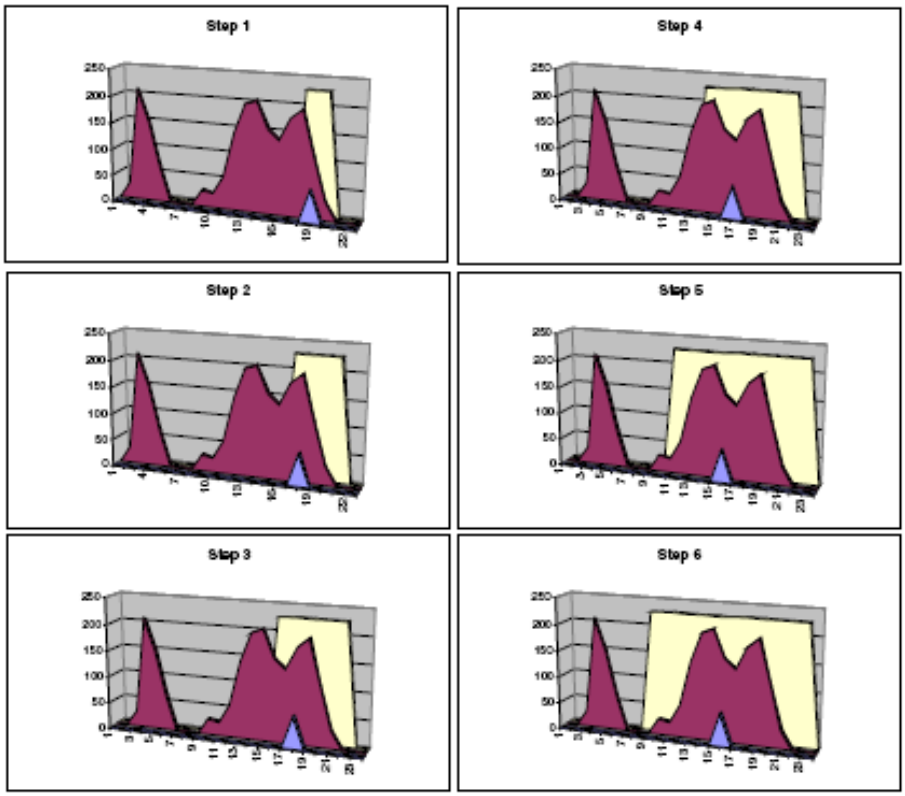

Figure 3-5. Enhancement of the tracking window by the time [12]

Rectangular CamShift window is shown behind the hue distribution, while triangle in front marks the window center. CamShift is shown iterating to convergence down the left then right columns. Starting from the converged search location in 4 bottom right, CamShift converges on new center of distribution in two iterations.

3.2.b.1.b Calculation of 2D Orientation

The *2D* orientation of the probability distribution is also easy to obtain by using the second moments in the course of CamShift operation, where the point (*x*,*y*) ranges over the search window, and *I*(*x*,*y*) is the pixel (probability) value at the point (*x*,*y*). Second moments are:

$$M_{20} = \sum_x \sum_y x^2 I(x,y)\,,\; M_{02} = \sum_x \sum_y x^2 I(x,y)\,.$$

Then the object orientation, or direction of the major axis, is:

$$\theta = \frac{\arctan\left(\dfrac{2\left(\dfrac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\dfrac{M_{20}}{M_{00}} - x_c^2\right) - \left(\dfrac{M_{02}}{M_{00}} - y_c^2\right)}\right)}{2}.$$

The first two eigenvalues, that is, length and width, of the probability distribution of the blob found by CamShift may be calculated in closed form as follows:

$$a = \frac{M_{20}}{M_{00}} - x_c^2,\ b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right), \text{ and } c = \frac{M_{02}}{M_{00}} - y_c^2.$$

Then length *l* and width *w* from the distribution centroid are:

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}},$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}}.$$

Figure 3-6. Motion detection with CAMShift

3.2.c Advantages of the Method

- Considers contour pixels only, insted of the whole frame

- Area with motion is set as the region of interrest, so the processing power is not used for the whole frame

3.2.d Disadvantages of the Method

- CAMShift tracks one window only and if there are two flames, with considerable distance from each other, one of them can be ignored.

3.3 FFT Method for Flame Detection

This method makes use of the contour detecting method explained in the previous chapter. It is a known fact that, the fire flame flickers at around 10 Hz, at it's border. It is possible to calculate the mean red value of the contours found and check if these values flicker, or in other words change their values with a frequency close to 10Hz. Presence of fire colored contours, changing their pixel values with a frequency is an indicator of fire.

3.1.1. Fourier Transform

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the *Fourier* or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image. The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression.

As we are only concerned with digital images, we will restrict this discussion to the *Discrete Fourier Transform* (DFT). The DFT is the sampled Fourier Transform and therefore does not contain all frequencies forming an image, but only a set of samples

which is large enough to fully describe the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image, *i.e.* the image in the spatial and Fourier domain are of the same size. For a square image of size N×N, the two-dimensional DFT is given by:

$$F(k,l) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\, e^{-i2\pi\left(\frac{ik}{N}+\frac{jl}{N}\right)}$$

where *f(i,j)* is the image in the spatial domain and the exponential term is the basis function corresponding to each point *F(k,l)* in the Fourier space. The equation can be interpreted as: the value of each point *F(k,l)* is obtained by multiplying the spatial image with the corresponding base function and summing the result.

The basis functions are sine and cosine waves with increasing frequencies, *i.e. F(0,0)* represents the DC-component of the image which corresponds to the average brightness and *F(N-1,N-1)* represents the highest frequency. In a similar way, the Fourier image can be re-transformed to the spatial domain. The inverse Fourier transform is given by:

$$f(i,j) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l)\, e^{i2\pi\left(\frac{ik}{N}+\frac{jl}{N}\right)}$$

To obtain the result for the above equations, a double sum has to be calculated for each image point. However, because the Fourier Transform is *separable*, it can be written as

$$F(k,l) = \frac{1}{N} \sum_{j=0}^{N-1} P(k,j)\, e^{-i2\pi\frac{jl}{N}}$$

where

$$P(k,j) = \frac{1}{N} \sum_{i=0}^{N-1} f(i,j)\, e^{-i2\pi\frac{ik}{N}}$$

Using these two formulas, the spatial domain image is first transformed into an intermediate image using *N* one-dimensional Fourier Transforms. This intermediate image is then transformed into the final image, again using *N* one-dimensional Fourier Transforms. Expressing the two-dimensional Fourier Transform in terms of a series of *2N* one-dimensional transforms decreases the number of required computations.

Even with these computational savings, the ordinary one-dimensional DFT has N^2 complexity. This can be reduced to $N \log_2 N$ if we employ the *Fast Fourier Transform* (FFT) to compute the one-dimensional DFTs. This is a significant improvement, in particular for large images. There are various forms of the FFT and most of them restrict the size of the input image that may be transformed, often to $N = 2^n$ where *n* is an integer. The mathematical details are well described in the literature.

The Fourier Transform produces a complex number valued output image which can be displayed with two images, either with the *real* and *imaginary* part or with *magnitude* and *phase*. In image processing, often only the magnitude of the Fourier Transform is displayed, as it contains most of the information of the geometric structure of the spatial domain image. However, if we want to re-transform the Fourier image into the correct spatial domain after some processing in the frequency domain, we must make sure to preserve both magnitude and phase of the Fourier image.

The Fourier domain image has a much greater range than the image in the spatial domain. Hence, to be sufficiently accurate, its values are usually calculated and stored in float values.

4. RESULTS

Results that we got from all three of the methods we have proposed are discussed and a comparison between the methods is explained in this section.

4.1. Results of the Frame Subtraction Method

The color detection algorithm, that is used in each of the three methods can sometimes detect pixels not belonging to a flame as fire-colored, and this is why it must be supported with detecting other properties of fire, like motion. Some ofthe problems observed with the frame subtraction method were, its false recognition of objects that have fire-like colors like skin or some gray objects, as flame. It's success is close to 100% when detecting fire but can also give negative false alarms, meaning it can detect fire even if there is no fire.

Since the frame subtraction method that is used for motion detection needs the program to calculate the difference for every pixel instead of determining a region of interest (ROI) it needs high processing power in order to implement it as a real-time application. CAMShift method for determining the ROI is proposed in the second method.

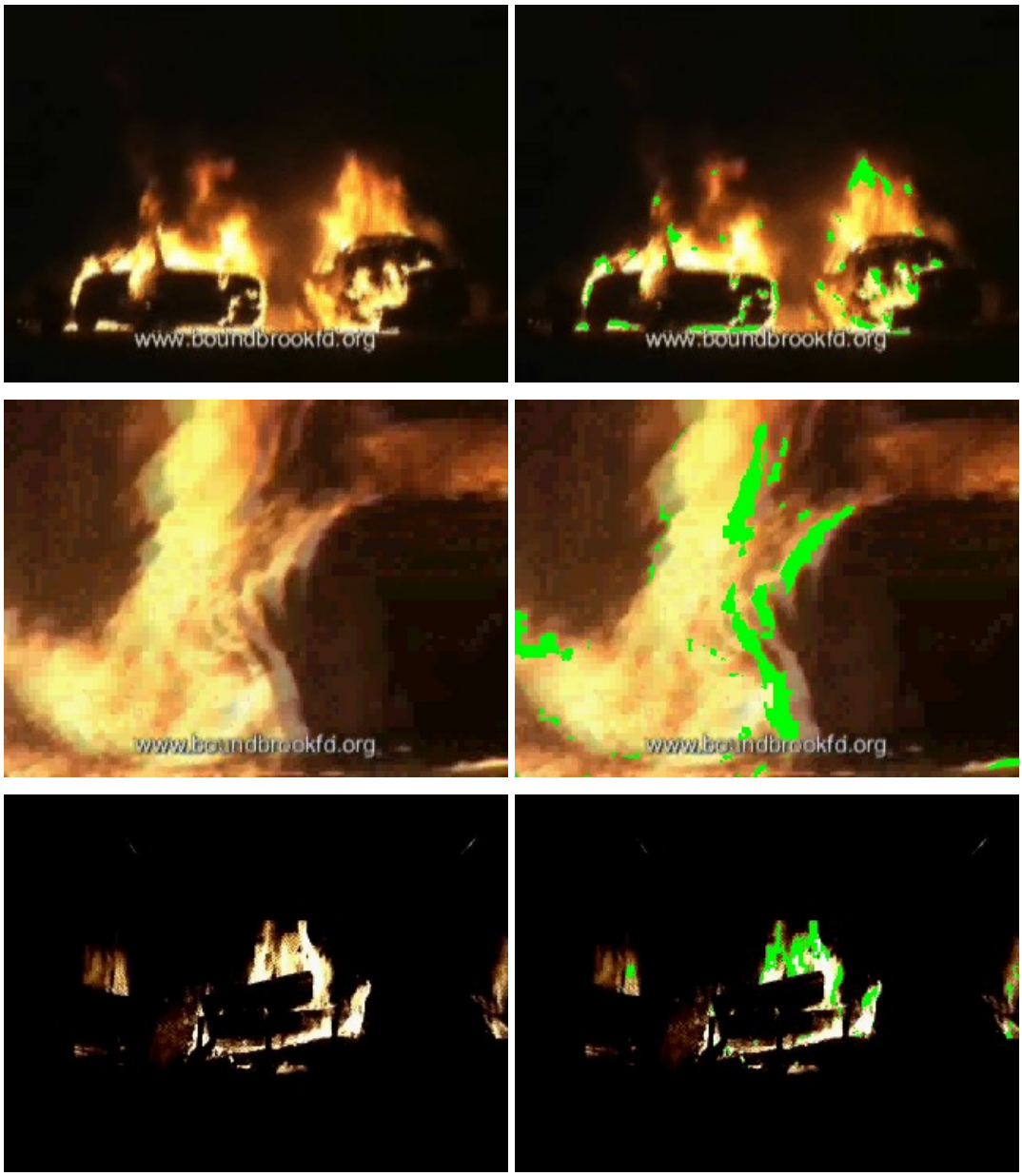

Figure 4.1. Results of the frame subtraction method (Continues next page)

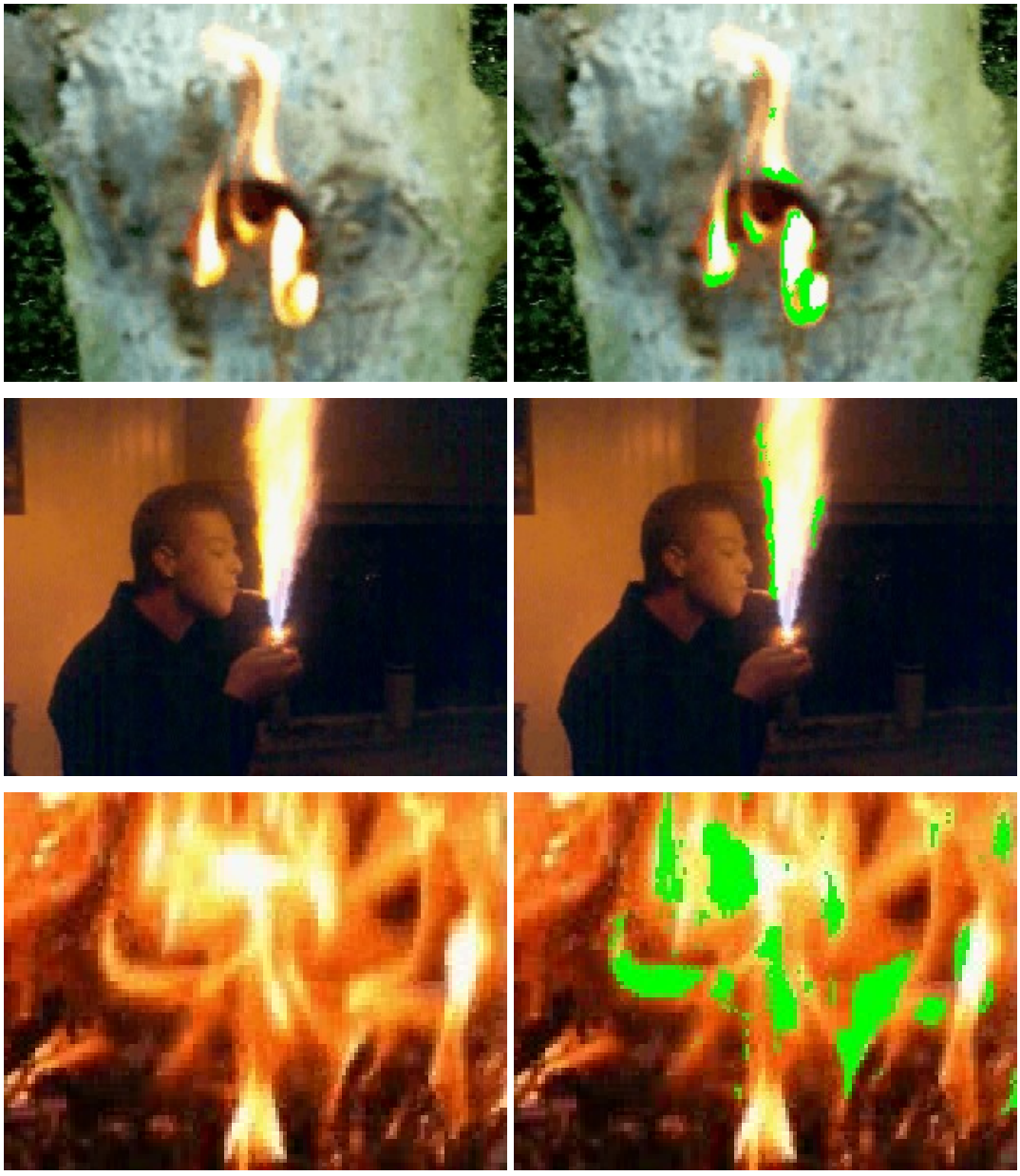

Figure 4.2. Results of the frame subtraction method (Continued from previous page)

4.2. Results of the CAMShift Method

Using of CAMShift algorithm for moving object detection causes an improvement of approximately 20-30% in speed.

In adition to reducing the number of the pixels used in calculations in order to detect motion, the number of fire-colored pixels used in flame detection are also reduced because only the contours of the objects are considered insted of the whole area of the objects.

In this method, appearance of a blue contour indicates the presence of the flame. For better view, Convex hull algorithm can be used in order to show the area where the contours are not closed, remembering that it would add to the processing performance of the system.

This method has similar performance at recognizing the flame object but needs less processing time and more reliable. It is also possible to increase the number of the objects being tracked with the CAMShift method.

Figure 4.3. Results of the CAMShift method (Continues next page)

Figure 4.4. Results of the CAMShift method (Continued from previous page)

4.3. Results of the FFT Method

Different than the first two methods, this method looks for a different property than color and motion of fire, which is flickering. This reduces the number of false alarms caused by flame-colored moving objects.

The system allows us to determine the threshold values of the flickering frequency and the acceptance value. In the resulting values of the DFT, if there is a value above the acceptance threshold (default: 1.0), among the values representing the frequencies above the frequency threshold (default: 8 Hz.), the investigated portion of the video (default: 10 Seconds) is detected as containing flame.

This method is effective in processing time because of making use of the FFT. It can be used with the previous method in order to detect three different properties of the flame (color, motion, flickering) and to determine the coordinates of the flame.

Results of this method, applied to a 249 frame video with 25 Fps can be seen in Figure 4.5. The result is an array with length of 128, where the last element of the array represents the highest frequency (12, 5 Hz.)

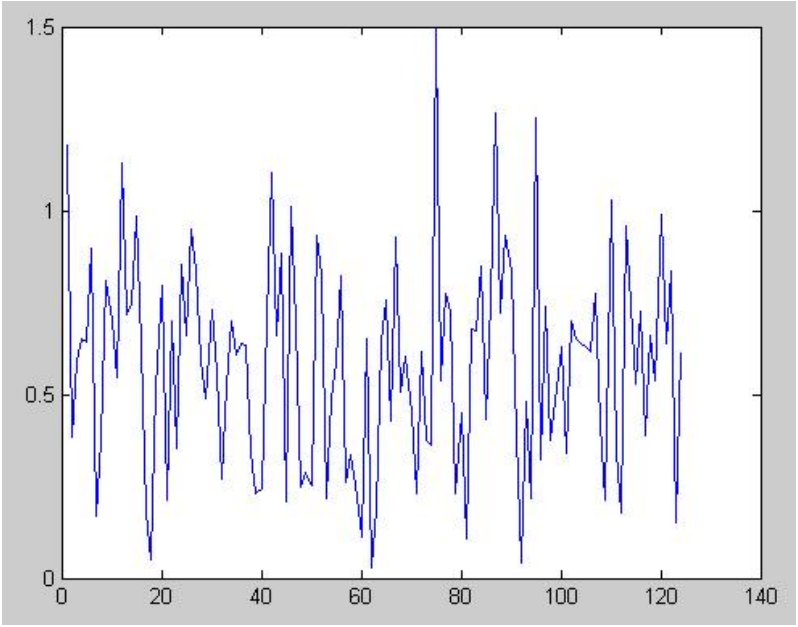

Figure 4.5. Results of the FFT method applied to a video sequence that includes flames

4.4. Comparison of All Three Methods

		Number of Fire Detected Frames			Number of False Positive Frames			Number of False Negative Frames			Description
	Number of Frames	Method1	Method2	Method3	Method1	Method2	Method3	Method1	Method2	Method3	
Video1	462	458	462	250	0	0	0	4	0	0	A burning car
Video2	250	250	250	250	0	0	0	0	0	0	Two burning cars
Video3	250	237	250	250	7	0	0	6	0	0	Fire in a market hand car
Video4	250	250	250	250	0	0	0	0	0	0	Flame in a fireplace
Video5	250	250	250	250	0	0	0	0	0	0	Fire with darkbackground
Video6	250	234	250	125	0	0	0	16	0	0	Flames made with spray and Sun coming out of the window
Video7	250	232	250	125	18	0	125	0	0	0	A moving hand lightening gas coming out of a pipe
Video8	250	250	250	250	0	0	0	0	0	0	A man firing woods with handmade flame gun

Table 4.1. Comparison of All Three Methods

5. REFERENCES

1- B. Ugur Töreyin, Yigithan Dedeoglu, Ugur Gudukbay, A. Enis Cetin , Computer Vision Based Method for Real-time Fire and Flame Detection, Pattern Recognition Letters, Elsevier

2- G. Healey, D. Slater, T. Lin, B. Drda, and D. Goedeke. *A system for real-time fire detection*. Computer Vision and Pattern Recognition, pages 605--606, 1993

3- Nobuyuki Fujiwara and Kenji Terada : Extraction of a Smoke Region Using Fractal Cording, IEEE International Symposium on Communications and Information Technologies 2004, No.28PM1F-01, (2004-10)

4- Wen-Bing Horng, Jian-Wen Peng, and Chih-Yuan Chen (2005), "A New Image-Based Real-Time Flame Detection Method Using Color Analysis," in *Proceedings of the 2005 IEEE International Conference on Networking, Sensing and Control*, Tucson, Arizona, USA, March 19-22, 2005

5- W. Phillips III, M. Shah, ve N. V. Lobo, Flame Recognition in Video, Pattern Recognition Letters, Volume 23 (1-3), sayfa 319-327, Jan. 2002

6- R. T. Collins, A. J. Lipton, T. Kanade, A System for Video Surveillance and Monitoring Proceedings of American Nuclear Society (ANS) Eighth International Topical Meeting on Robotics and Remote Systems, Pittsburgh, PA, 25-29 Apr. 1999

7- C. B. Liu ve N. Ahuja, Vision Based Fire Detection, IEEE International Conference on Pattern Recognition, Cambridge, UK, Aug. 2004

8- Wavelet Based Real-Time Smoke Detection in Video, B. Ugur Töreyin, Yigithan Dedeoglu, A. Enis Cetin, EUSIPCO 2005, Antalya, Turkey.

9- T. Chen, P. Wu, and Y. Chiou, “An early fire-detection method based on image processing,” in ICIP ’04, 2004, pp. 1707–1710

10- Fastcom Technology SA, Method and Device for Detecting Fires Based on Image Analysis, Patent Coop. Treaty (PCT) Pubn.No: WO02/069292, Boulevard de Grancy 19A, CH-1006 Lausanne, Switzerland, 2002

11- H. Yamagishi and J. Yamaguchi, Fire Flame Detection Algorithm Using a Color Camera , Sogo Keibi Hosho Co., Ltd., Japan

12- Intel Corporation, Open Source Computer Vision Library Refernce Manual

6. SOURCES

- Open Source Computer Vision Library Refernce Manual
- http://tech.groups.yahoo.com/group/OpenCV/
- Smith SW, The Scientist and Engineer's Guide to Digital Signal Processing 2nd Ed,
- Shapira LG, Computer Vision, Prentice Hall
- http://www.cee.hw.ac.uk/hipr/html/canny.html
- http://doc.trolltech.com/4.0/index.html
- http://www.fftw.org
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm

7. RESUME

Name Surname	: Eren Aykın
Date of Birth	: June 21st 1983
Place of Birth	: İstanbul
Highschool	: Kdz. Ereğli Anadolu Lisesi
Internships	: Mikrosay Yazılım A.Ş. / İstanbul
	ISDEMİR A.Ş. Bilgi İşlem / İskenderun
	Microsoft Yaz Okulu / İstanbul
	YTU Yazılım Lab. / İstanbul
	Imagine Web Solutions / Kazakhstan